Jacobs University Bremen

CO-522-B Communications Basics Lab

Fall 2021

Lab Experiment 2: Receiver Simulink Study

Priontu Chowdhury

Introduction

In this lab, we built on the knowledge of the transmitter system, and studied the operation of the receiver subsystem using Simulink. We used the transmitter as a starting point, and built a receiver in this lab, which formed a complete communications system together. We were introduced to important concepts such as:

Down conversion: We shifted up the signal to a convenient transmit frequency at the transmitter. At the receiver, we must shift it back to baseband frequency. We accomplish this by multiplying it with the conjugate of the carrier used for up-conversion.

Matched filtering: We design a filter that provides an optimal SNR in order to simulate a real scenario. The best choice is to use a filter identical to the transmit pulse-shaping filter, which is we call a matched filter. We can get a raised cosine response having the zero ISI property by using a root raised cosine filter in the transmitter and receiver. This way, the cascaded response is a raised cosine filter and both transmitter and receiver have identical matched filters.

Decision rules: The decision block decides what symbol was most likely transmitted based on the received I/Q sample. For lower-order modulation, this is usually comparing the value to a threshold.

Carrier recovery: Transmitting signals wirelessly makes it extremely difficult to make sure that the transmitted and received signals are the same. There could be an offset that ranges from 10 kHz to 100 kHz in wireless RF systems, which creates significant error in the receiver chain. To remove the carrier offset, we recover the carrier by removing modulation and lock an oscillator to the signal.

Symbol recovery: The symbol clocks at transmitter and receiver are not the same, and we don’t know how much relative delay there is between transmitter and receiver. We implement Symbol Timing Recovery by multiplying the baseband with a copy of itself shifted by half of the symbol time. This creates a sinusoid that has a frequency equal to the symbol rate, where zero-crossings of the sinusoid happen at optimal sample points. A PLL is then used to generate a sine wave locked to this carrier. The delay block allows the timing to be slightly adjusted as needed. We check for positive zero crossing of the sine wave, which is where we sample.

Execution

1. Creating a Subsystem

The following transmitter was developed in the previous lab:

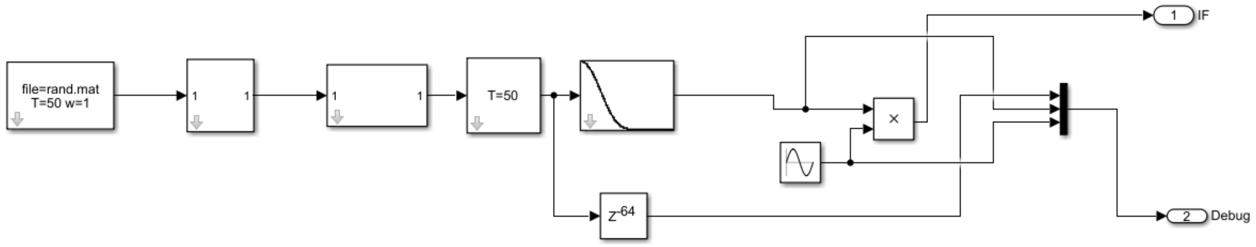


Figure 1: BPSK Transmitter

The following subsystem is created using the above transmitter:

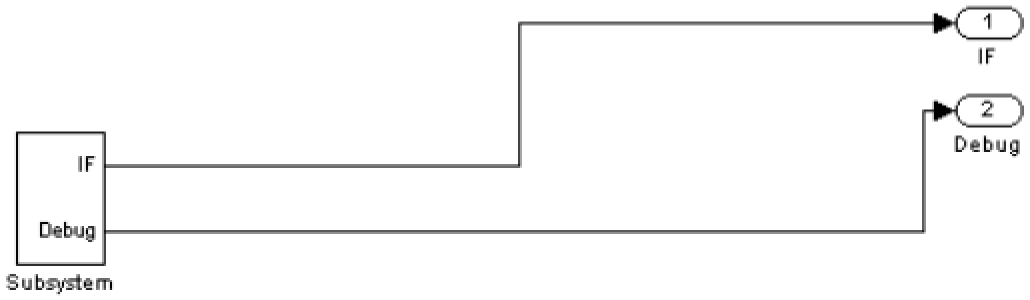


Figure 2: BPSK Transmitter Subsystem

2. Basic BPSK Receiver

We now create a basic BPSK receiver as shown below:

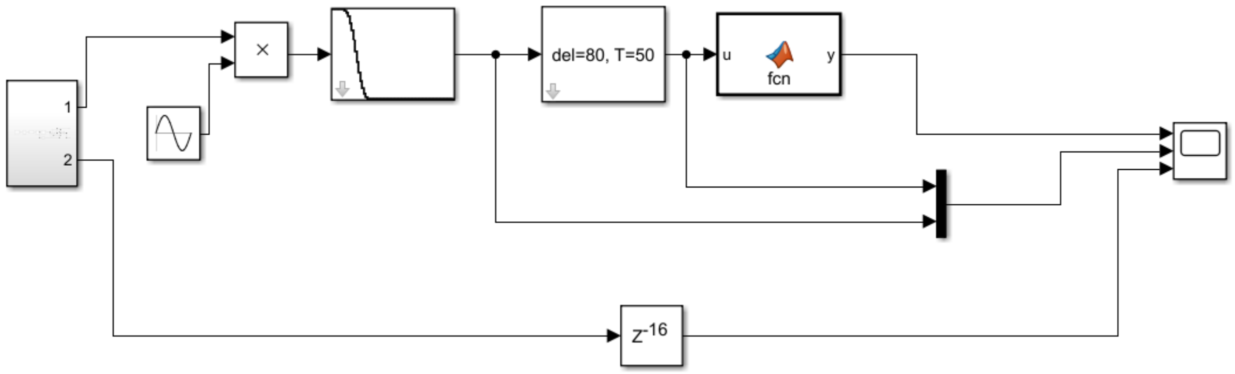


Figure 3: A Basic BPSK Receiver

The function block (fcn) is provided the following functionality:

```
function y = fcn(u)
if (u > 0)
    y = 1;
else
    y = 0;
end
```

On running the simulation, we see the following results on the scope:

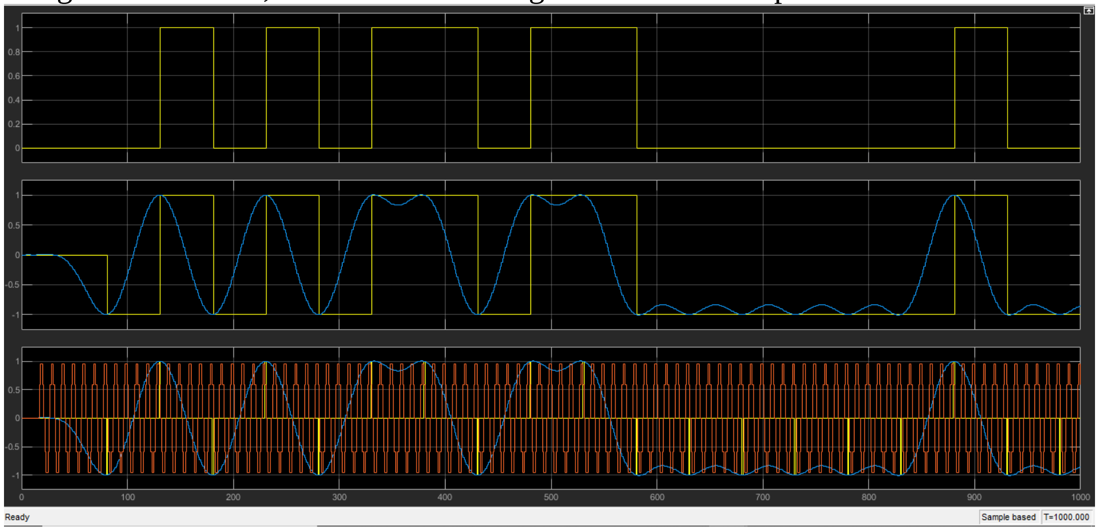


Figure 4: Simulation results

The bottom window contains the inputs (sine carrier in red, sampler output in yellow, and RC Filter output in blue), the middle window contains the low-pass filtered (blue) and sampled (yellow) results, and the top window contains the output after running the sampled data through the Matlab Function block.

3. Matched Filtering

We obtain optimal noise performance if the receive filter is the same as the transmit filter. Consequently, we modify our BPSK design and use the same raised cosine filter as in the transmitter. We also change the delay of the signals for sampling since this filter gives a 64 sample delay. The final structure is shown below:

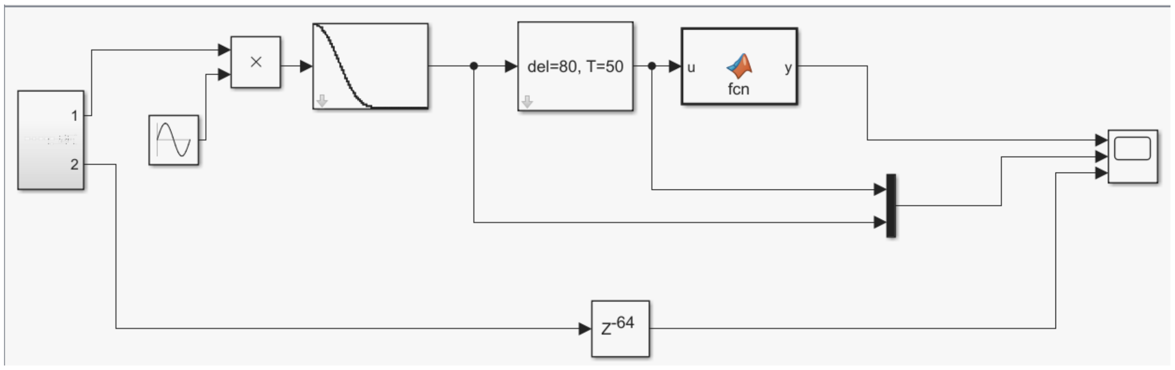


Figure 5: Receiver System with Matched Filter

Simulation results for this system is provided below:

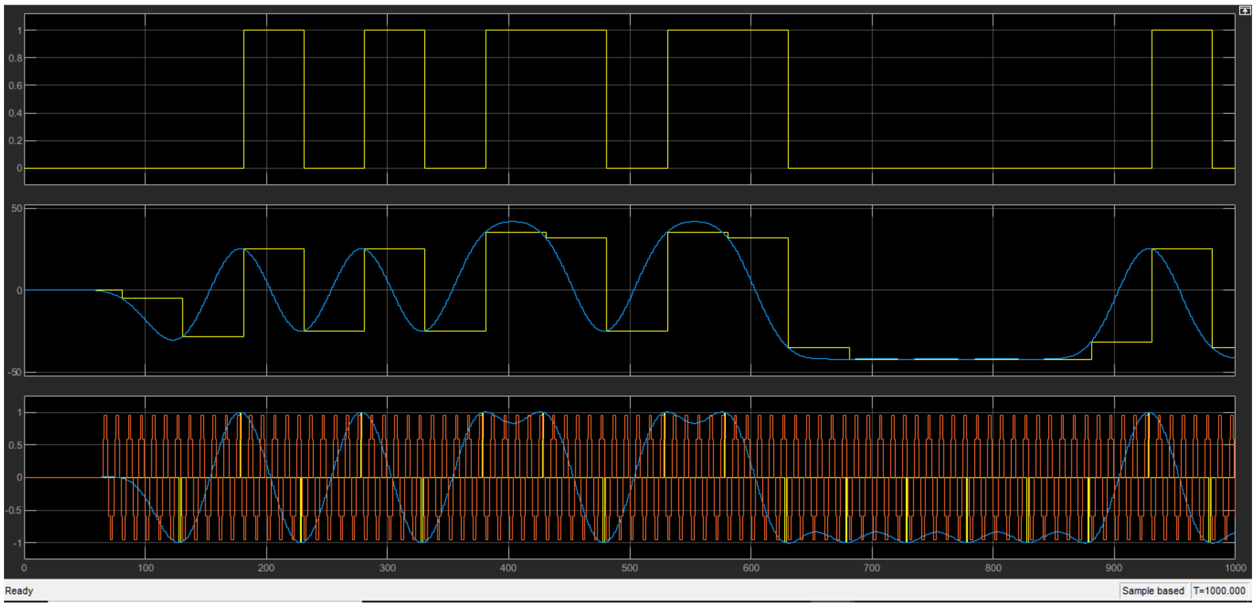


Figure 6: Simulation results for above system

4. Root Raised-Cosine Matched Filter

One problem with the root raised cosine (RRC) filter is that the time response is much longer than the RC filter.

So, to support the RRC filter we need to do the following:

1. Change the transmit and receive filters to root filters.
2. Make the filters 512 taps long to support the longer time response
3. Since each filter now creates 256 samples of delay, make changes to any delay blocks that depend on this.

The final system looks as follows:

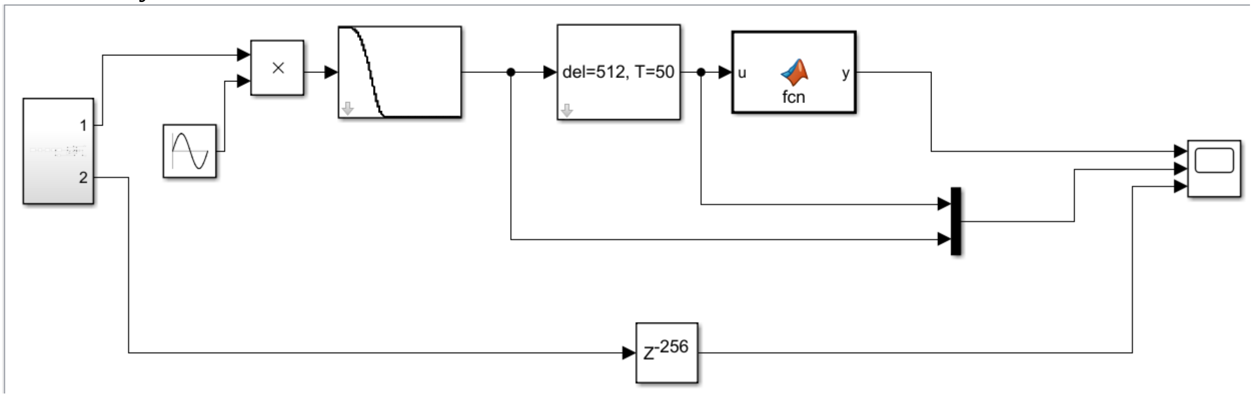


Figure 7: System modified to facilitate RRC response

Similar modifications were made at the transmitter:

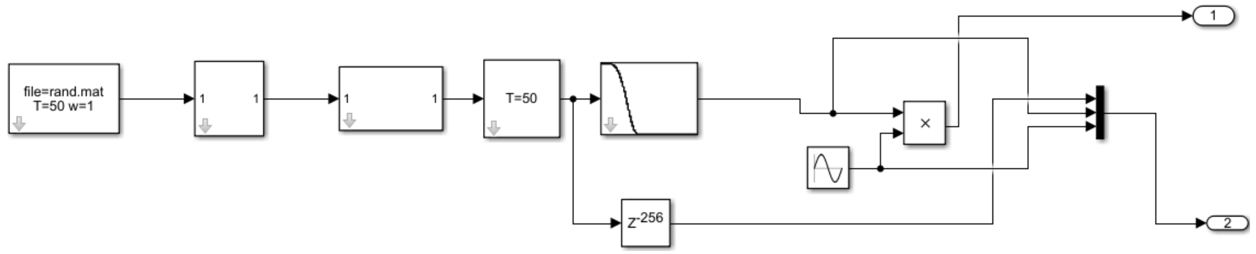


Figure 8: BPSK Transmitter modified for RRC response

The simulation results for the system are provided below:

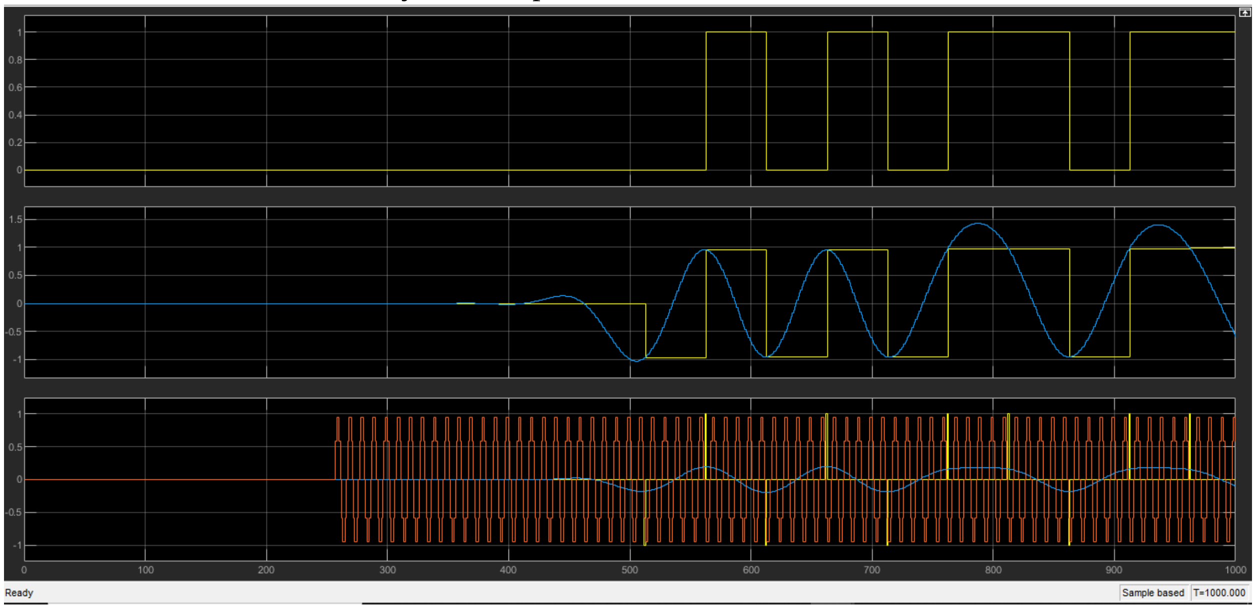


Figure 9: Simulation Results for above system

5. Carrier Recovery

In a practical communications system, the receiver oscillator will not be the same as the transmit one. To see the effect of carriers that are not the same, we modify the transmitter so that the sin() block is driven with a modified clock as depicted below:

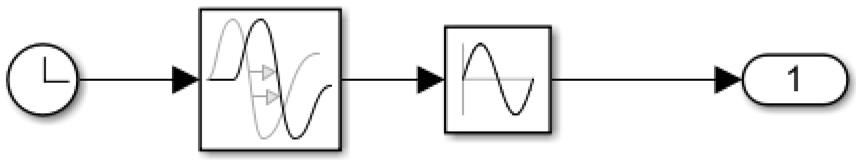


Figure 10: Sine block driven with a modified clock

To make it easy for the receiver to recover the carrier, we will add a pilot tone to the transmit signal, which is simply an unmodulated component that in this case has half the frequency of our carrier. The receiver can then isolate this tone and double the frequency to get the carrier back. Any delay or frequency shift in the transmit carrier will be completely reflected in the pilot tone, allowing the receiver to track these differences.

The final transmitter system looks as follows:

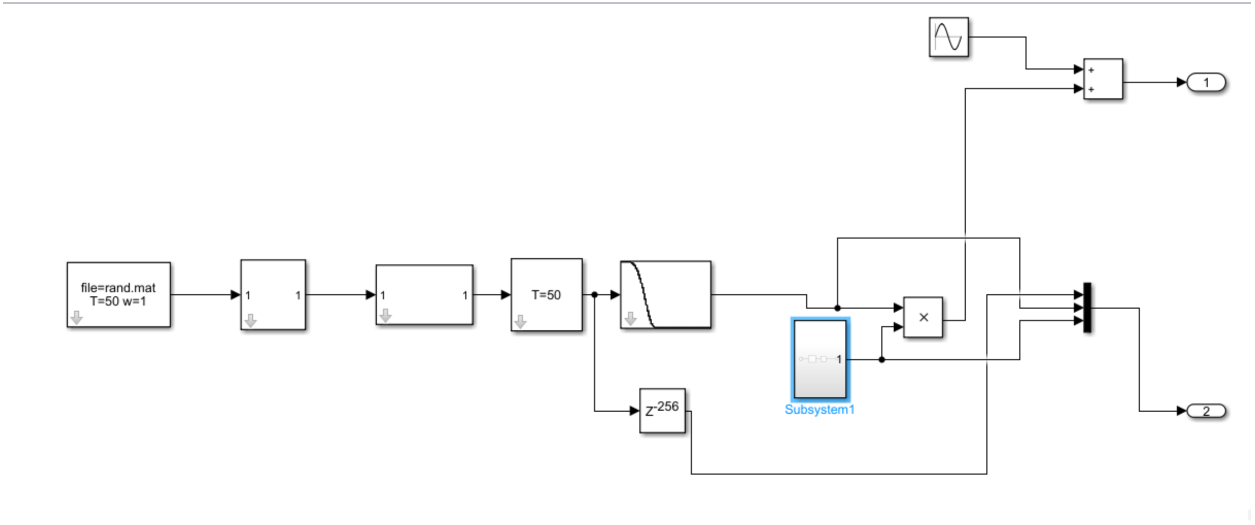
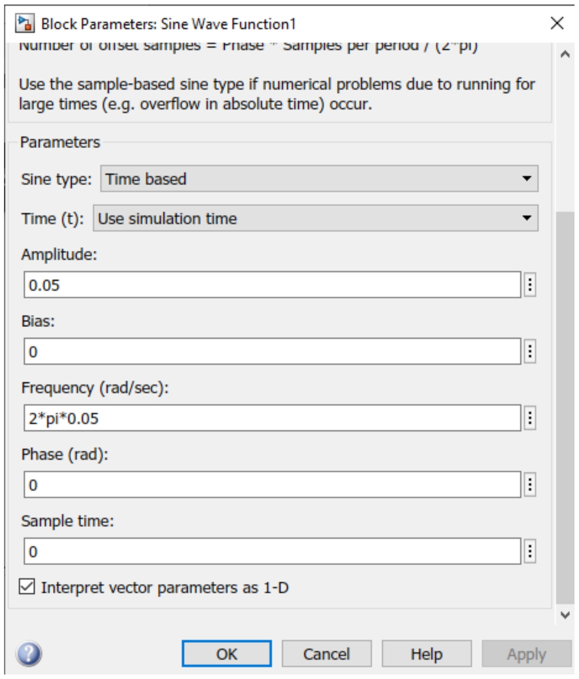


Figure 11: Transmitter system with modified sine block

The sine wave (pilot tone) that is added to the output signal has the following settings:



Number of offset samples = Phase * Samples per period / (2*pi)

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 0.05

Bias: 0

Frequency (rad/sec): 2*pi*0.05

Phase (rad): 0

Sample time: 0

☑ Interpret vector parameters as 1-D

Figure 12: Pilot Tone Settings

In the receiver, we need to first isolate the pilot tone using a narrow bandpass filter. For this, we use a RECT filter.

The receiver looks as follows after all the required modifications:

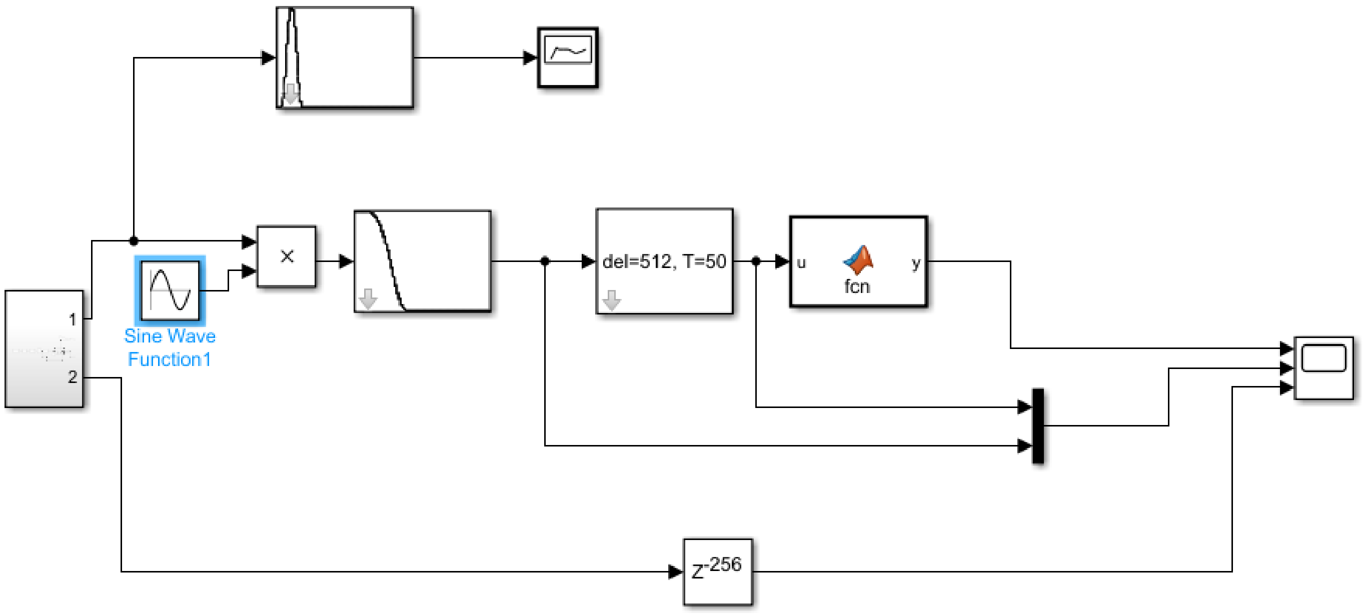
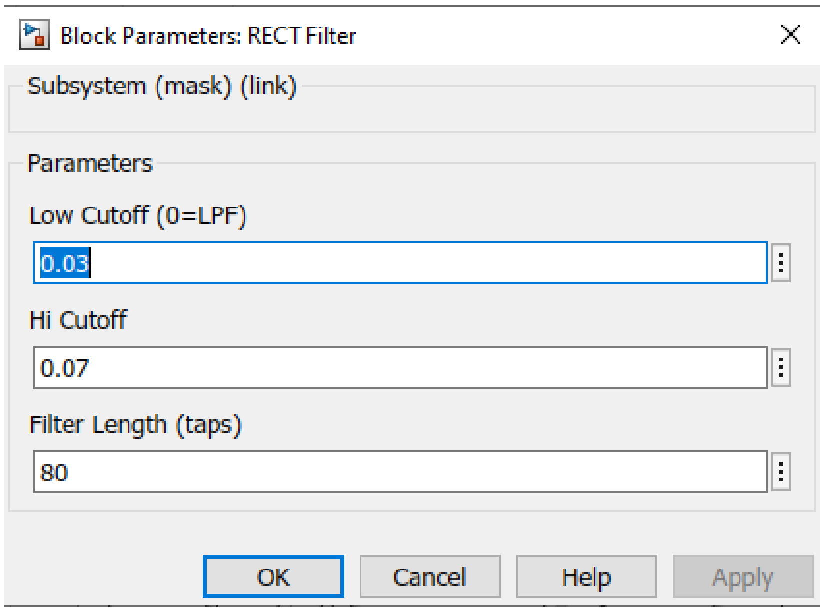


Figure 13: Modified Receiver System for Carrier Recovery

The RECT filter settings for the carrier recovery branch are provided below:


Block Parameters: RECT Filter

Subsystem (mask) (link)

Parameters

Low Cutoff (0=LPF): 0.03

Hi Cutoff: 0.07

Filter Length (taps): 80

Figure 14: Filter Settings for Carrier Recovery Branch

The simulation results are provided below:
Scope Results:

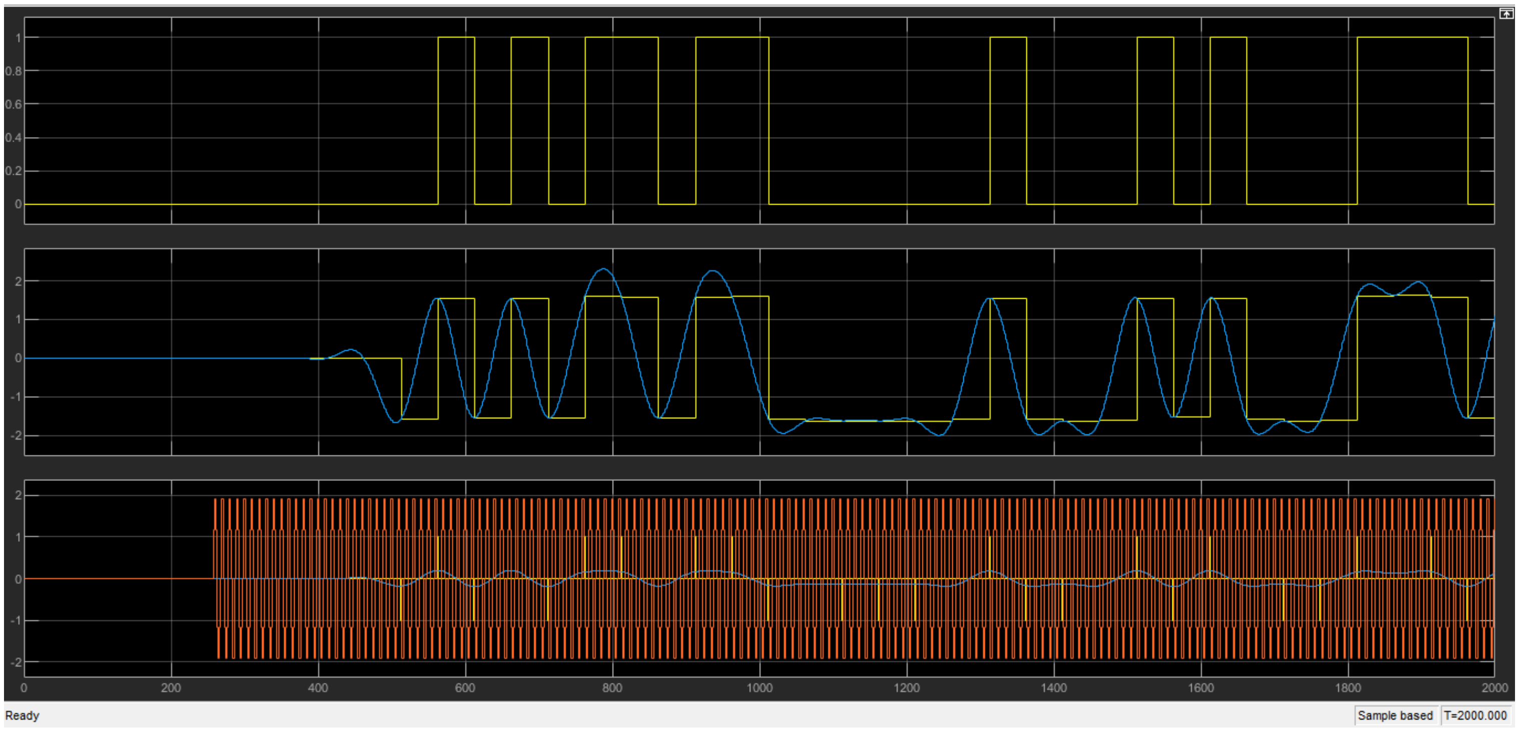


Figure 15: Simulation Results from Receiver

PSD Results:

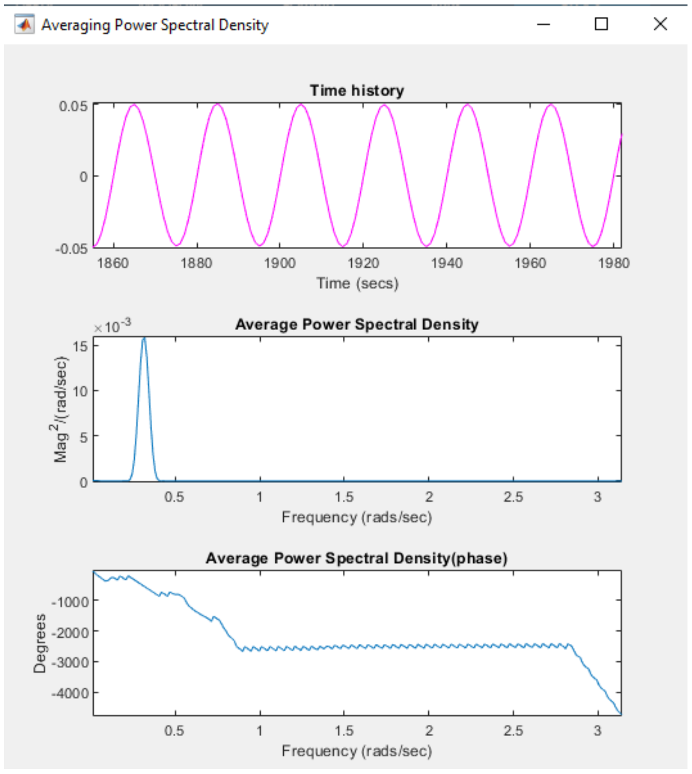


Figure 16: PSD Results – Isolated Pilot

6. PLL-based Synchronization

A PLL is a device that takes a sine wave as input and tracks its phase using a loop. It generates its own sine wave internally, whose phase is locked to the incoming signal. The PLL is also free to generate any other clock or sine wave signals that are needed based on this phase.

To understand how the PLL works, we create a new empty design and put the PLL block in it with a source. We also change the model settings as follows:
Solver selection – fixed step
Solver – discrete

The simple PLL system is shown below:

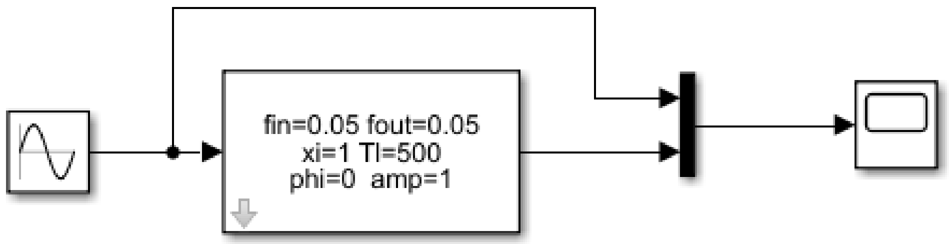


Figure 17: Simple PLL based system

The simulation results are provided below:

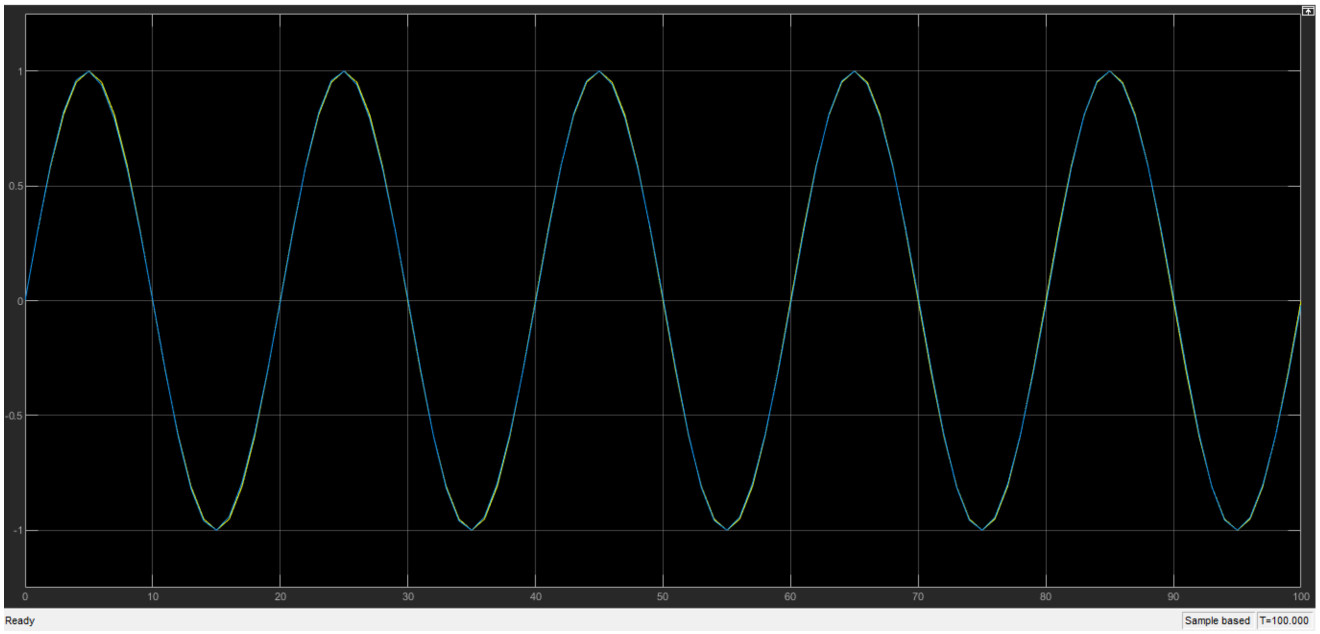


Figure 18: PLL Simulation results

We can see that the input and output have the same phase as a result of the PLL.

Now, we add the PLL to our design and provided the settings as instructed. The modified system is shown below:

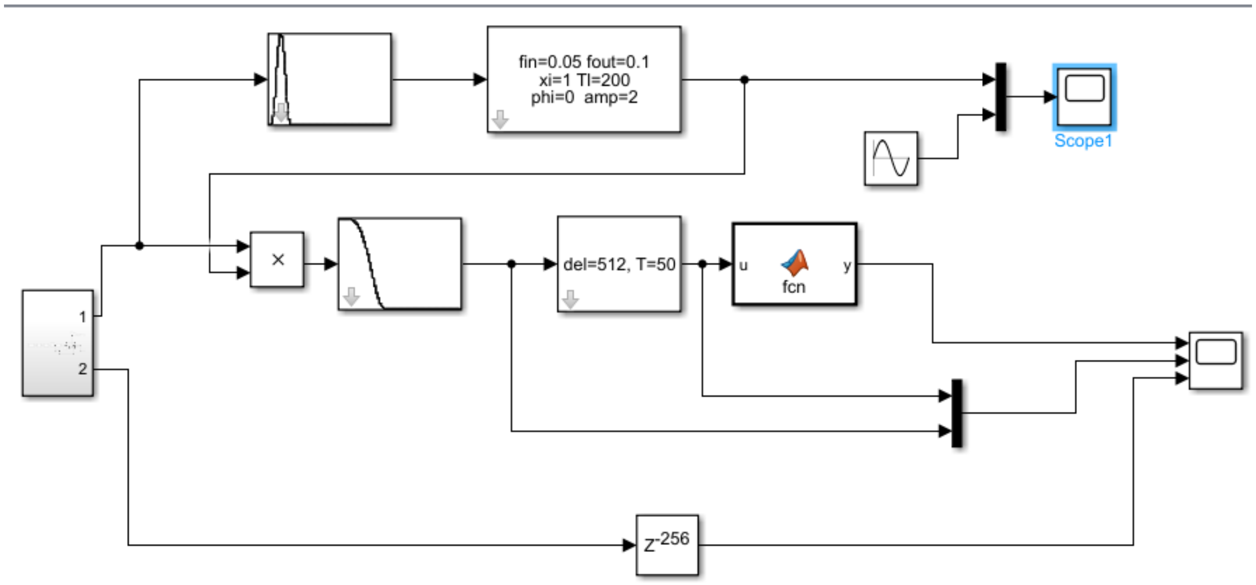


Figure 19: Receiver System with PLL block

The results from the first scope are provided below:

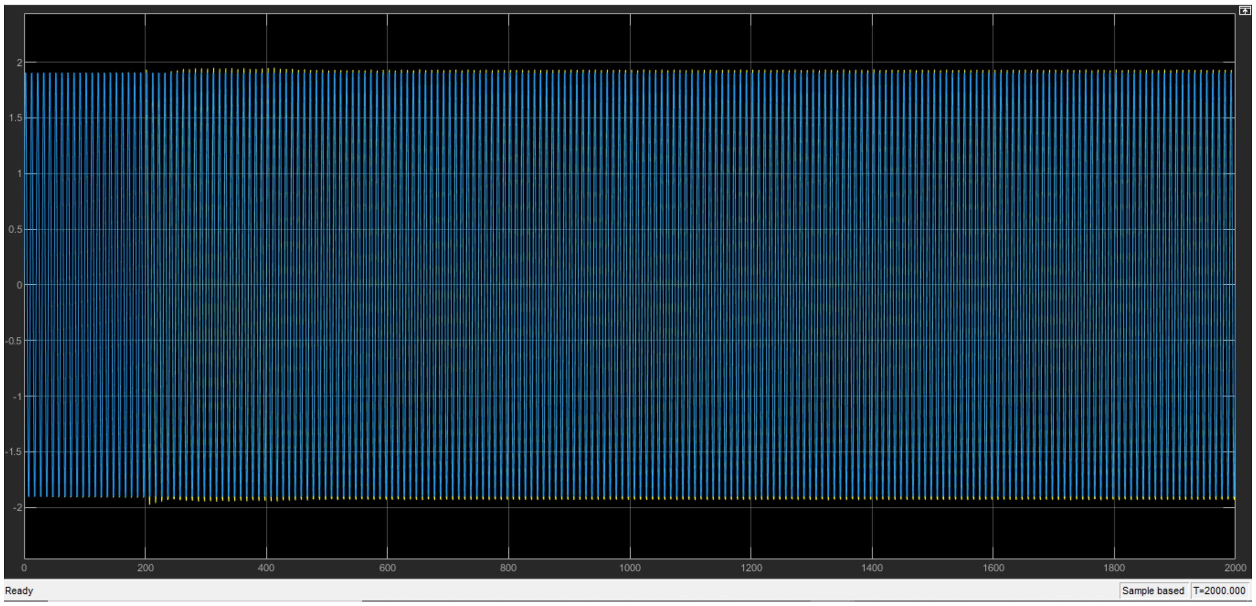


Figure 20: Output of PLL

We can see that the pilot tone’s phase has been corrected.

The results from the second scope are provided below:

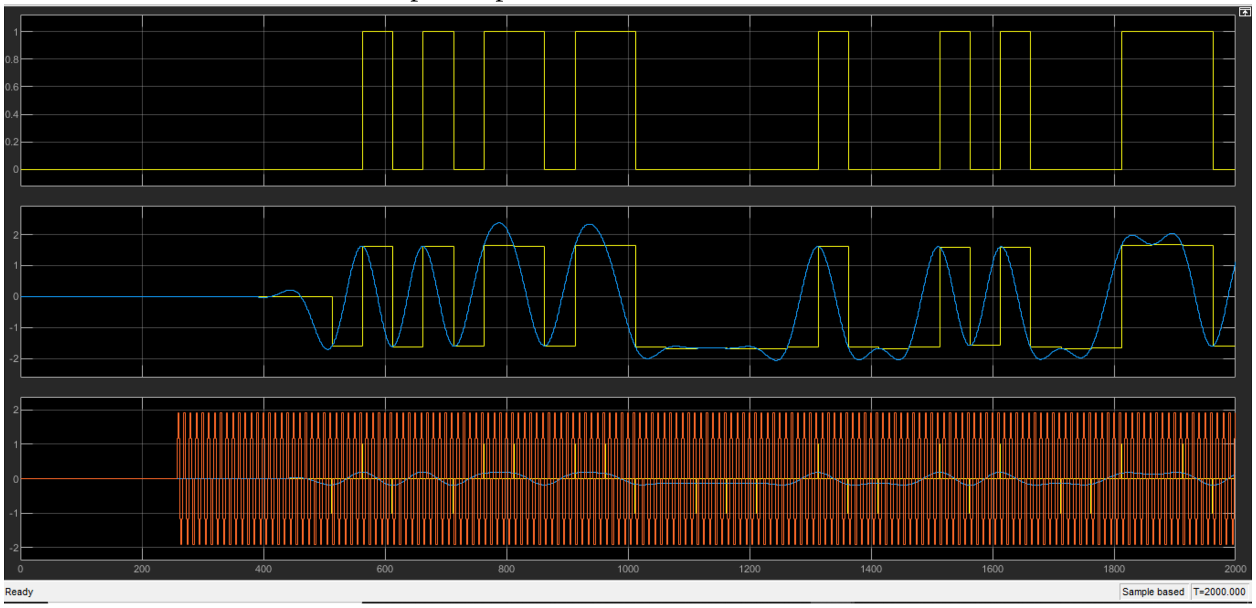


Figure 21: Simulation results for the modified system

7. Symbol Timing Recovery

We wish to sample the matched-filtered system at optimal points. We add the Symbol Timing Recovery block and the Sampler block and provide the required settings as instructed. The final system is provided below:

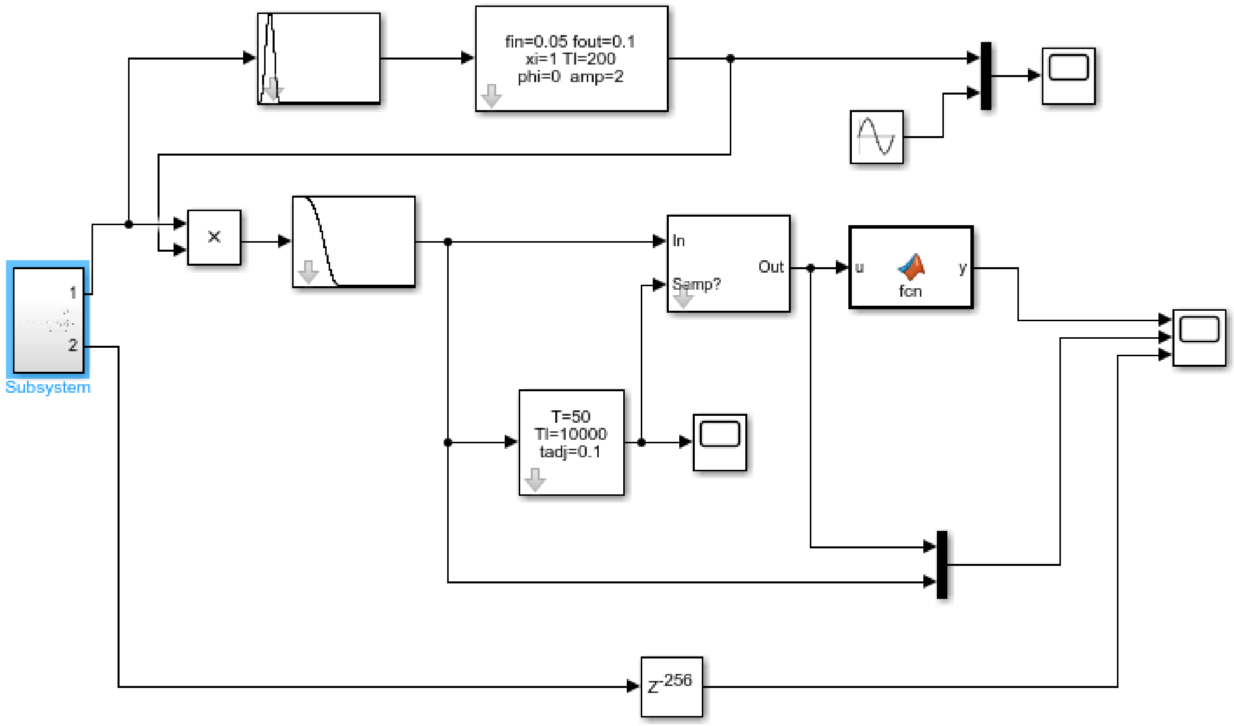


Figure 22: Modified System for Symbol Timing Recovery

The system works by multiplying the baseband signal with a copy of itself shifted by half of the symbol time. This creates a sinusoid that has a frequency equal to the symbol rate, where the zero-crossings of the sinusoid occurs at optimal sample points. The PLL is then used to generate a sine wave locked to this carrier. Then we check for a positive zero crossing of the sine wave, which is where we sample. This creates a train of pulses at the optimal sample points.

The following results are for 10K samples:

The PLL output is provided below:

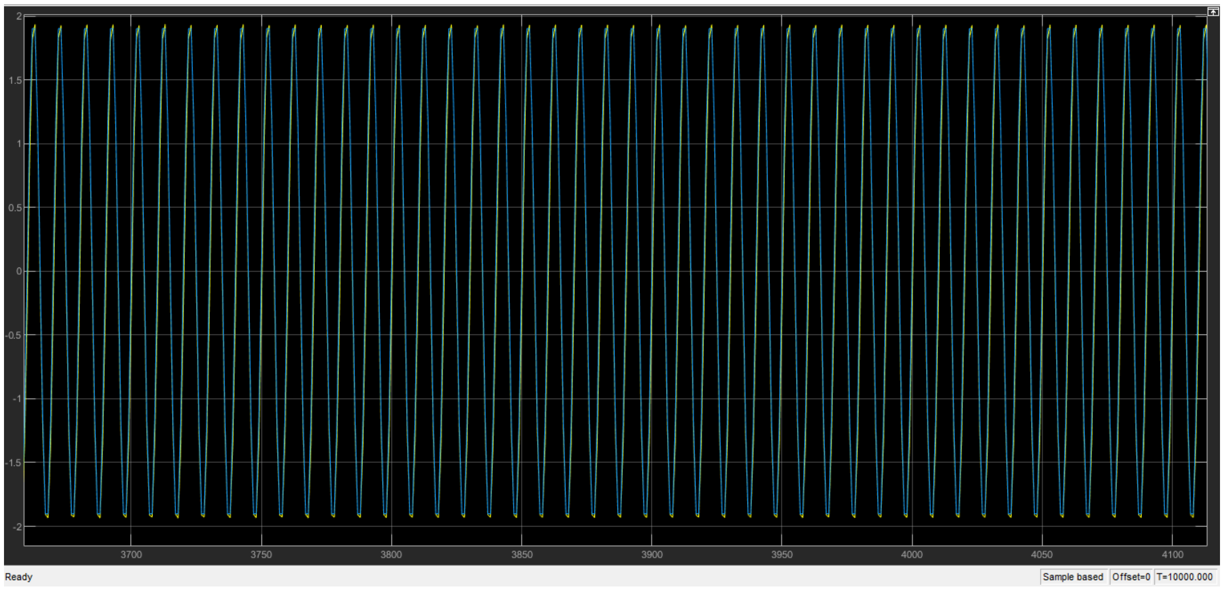


Figure 23: PLL Output

The output of the system timing recovery block is shown below:

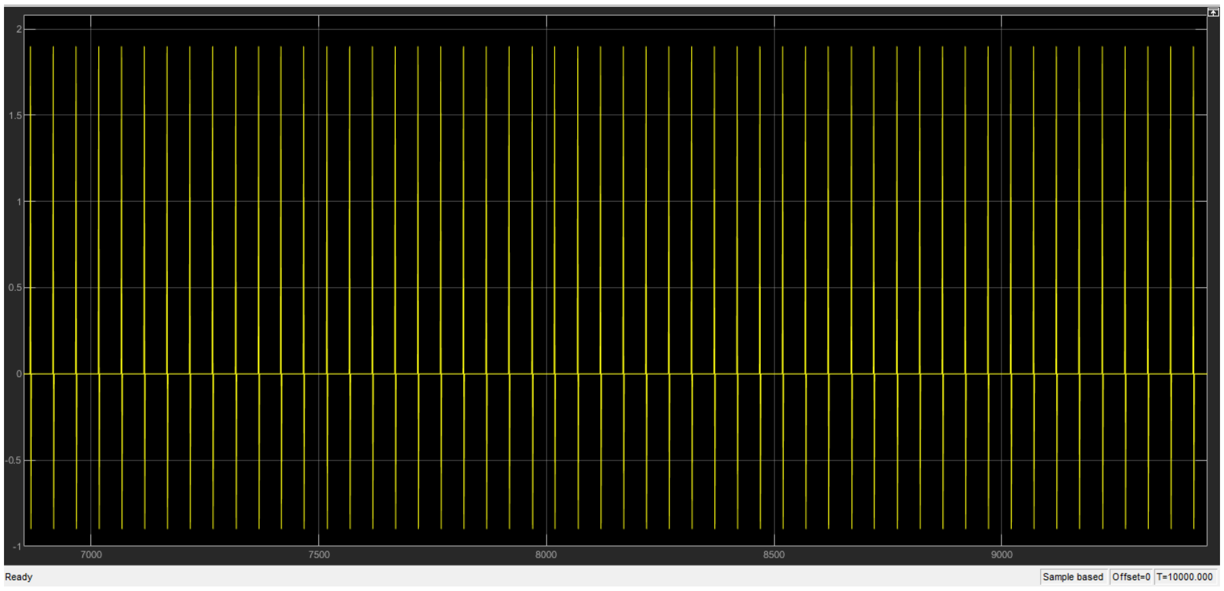


Figure 24: System Timing Recovery Output

The system response for the receiver is shown below:

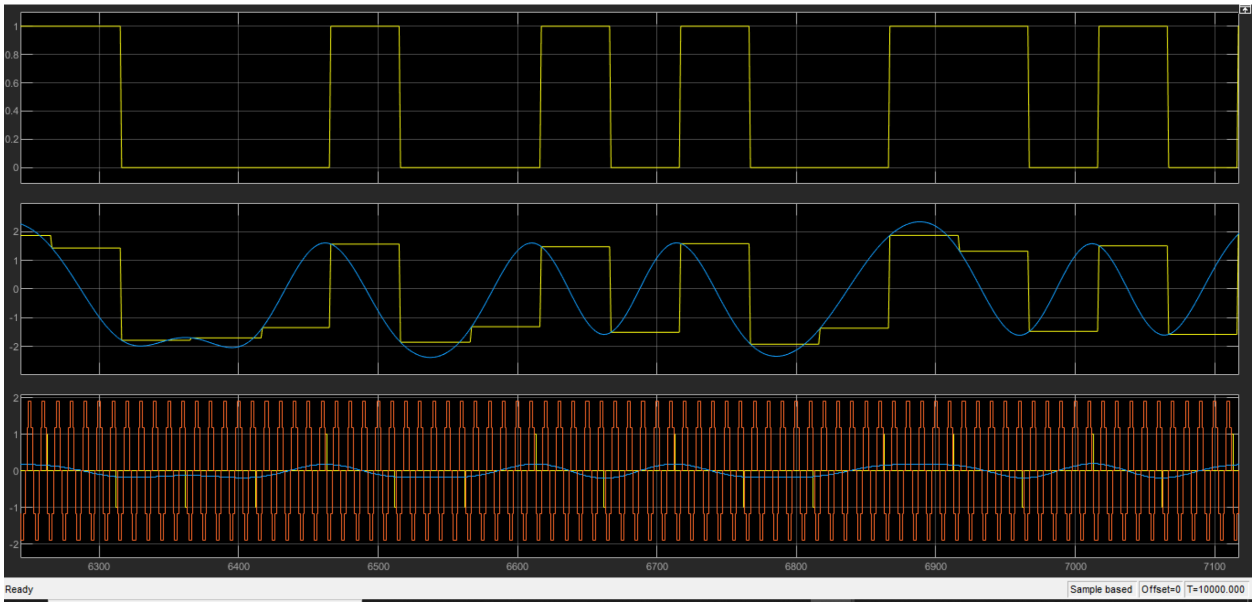


Figure 25: Receiver system output

The following results are for 200K samples:

The PLL output is provided below:

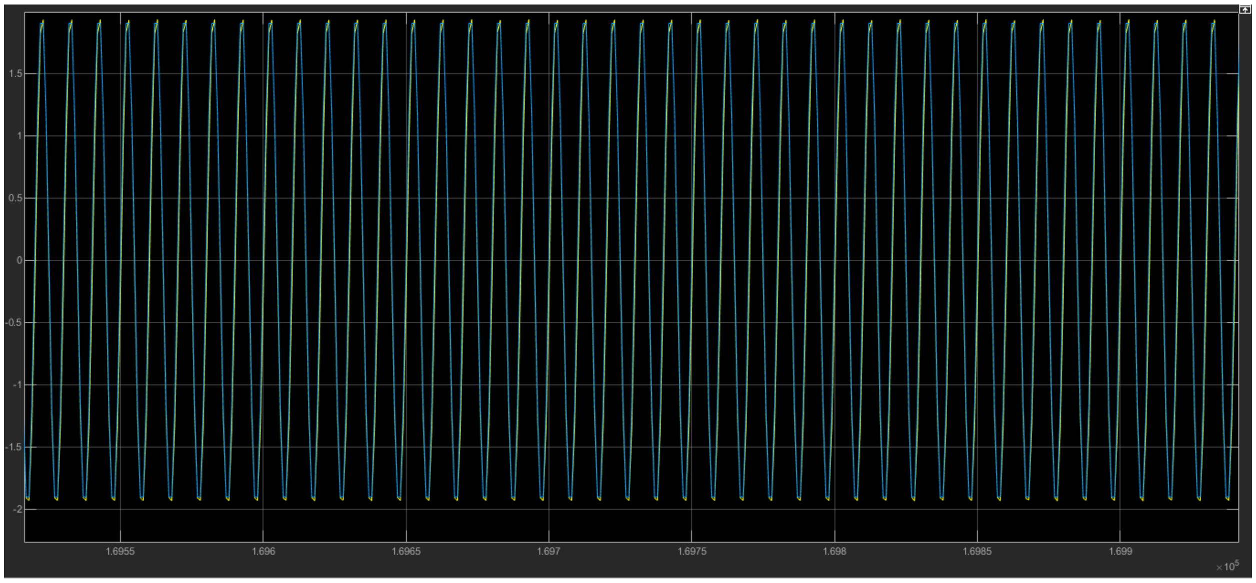


Figure 26: PLL Output

The output of the system timing recovery block is shown below:

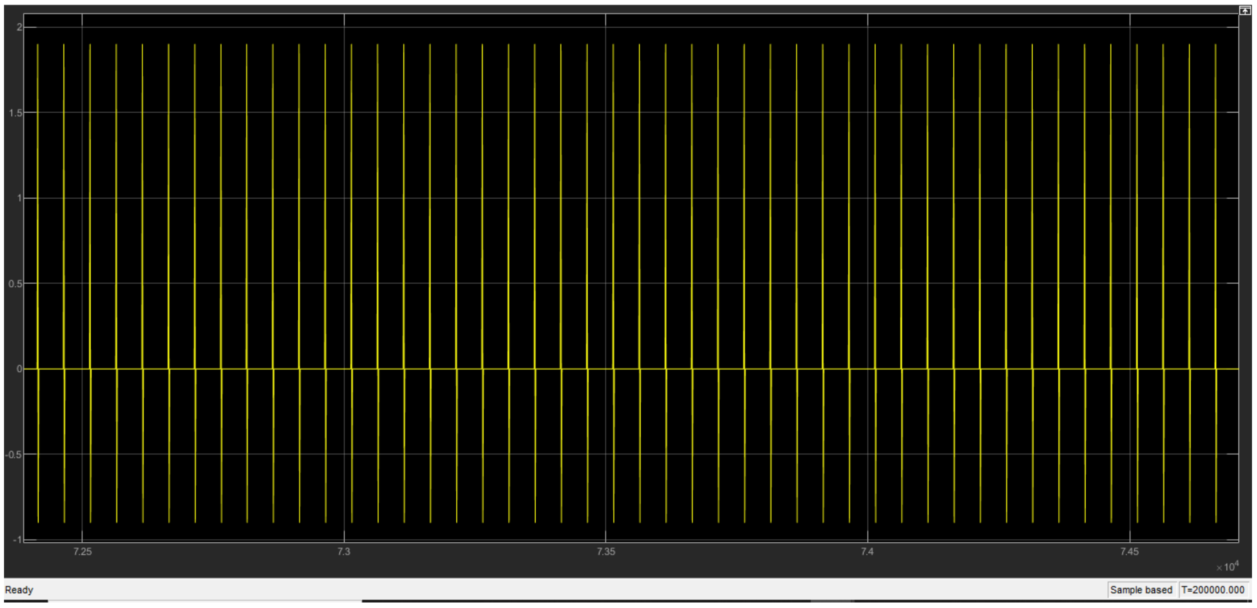


Figure 27: System Timing Recovery Output

The system response for the receiver is shown below:

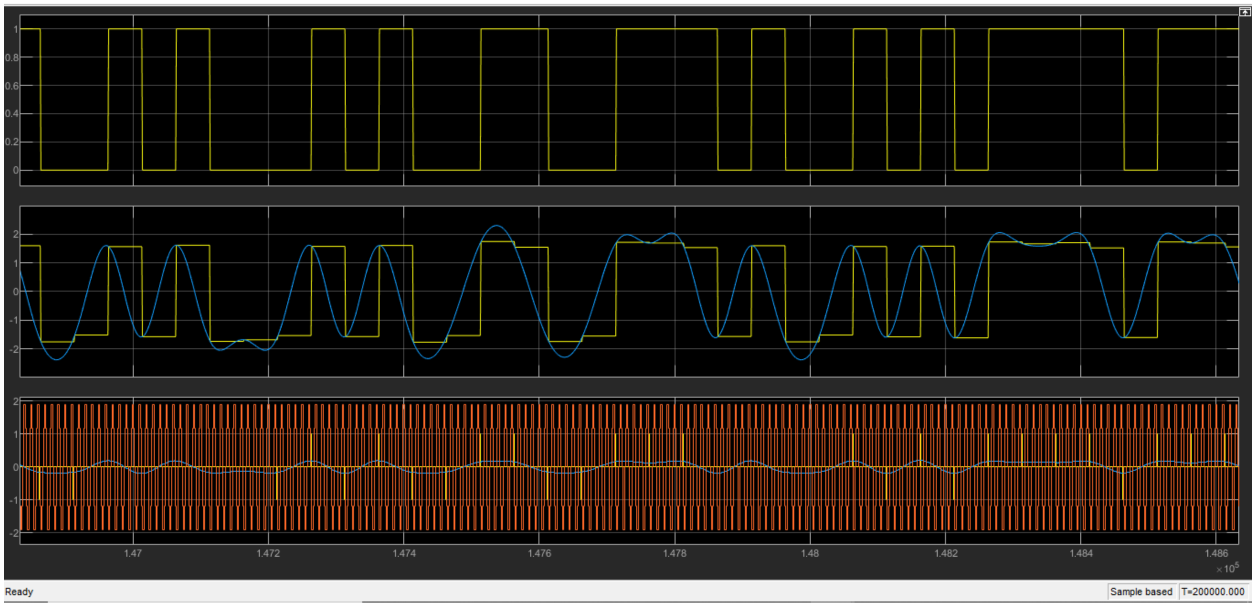


Figure 28: Receiver system output

8. Noise

For this part, we add a small amount of noise to our system. The system is shown below:

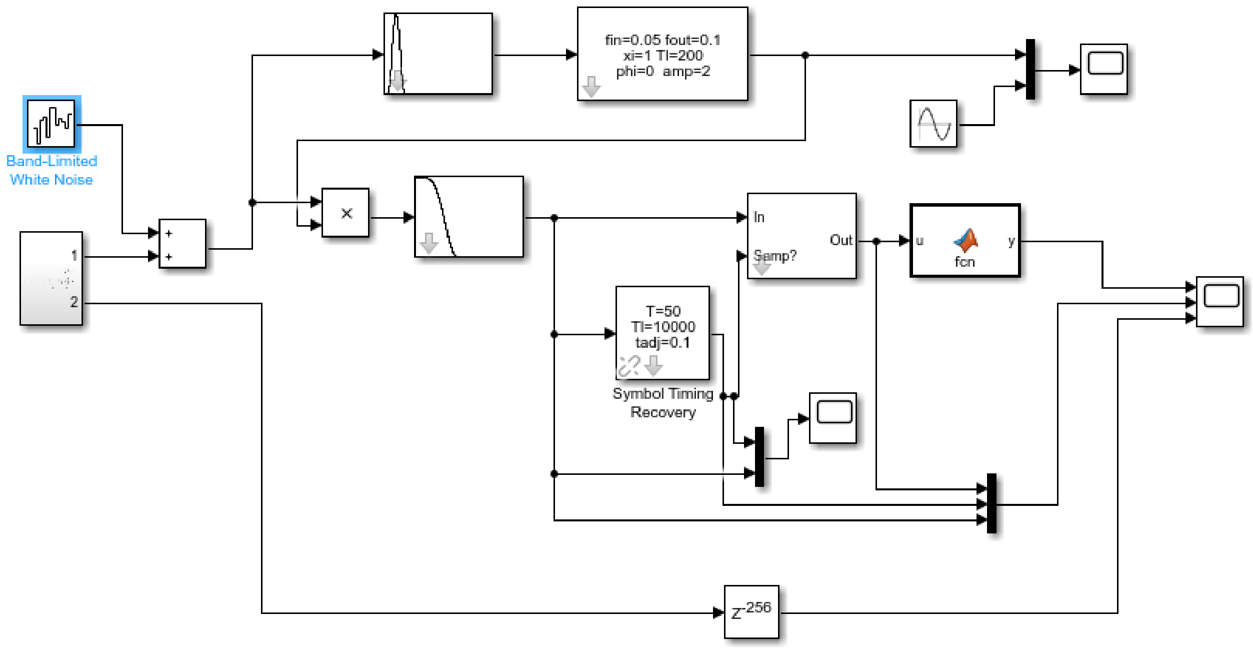
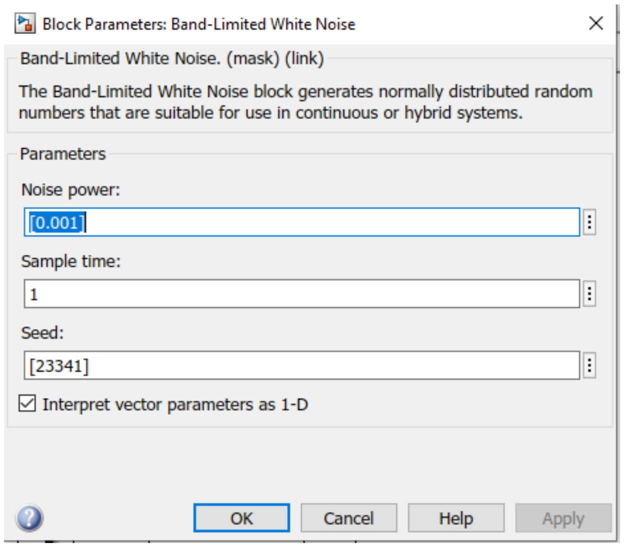


Figure 29: System with Noise

The noise block is provided the following settings:



Block Parameters: Band-Limited White Noise

Band-Limited White Noise. (mask) (link)

The Band-Limited White Noise block generates normally distributed random numbers that are suitable for use in continuous or hybrid systems.

Parameters

Noise power: [0.001]

Sample time: 1

Seed: [23341]

☑ Interpret vector parameters as 1-D

Figure 30: Noise block settings

This gives an SNR out of the matched filter of about 15 dB.

The simulation results are provided below:

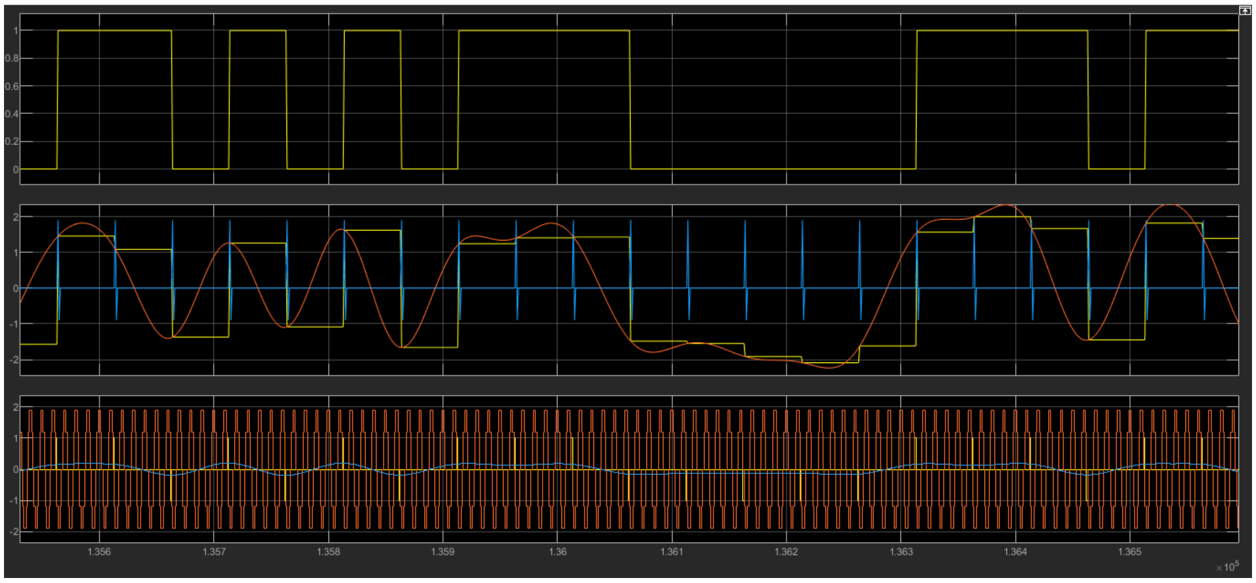


Figure 31: Receiver System Output (Noise Power: 0.001)

Now, we change the noise power to 0.01 and rerun the simulation:

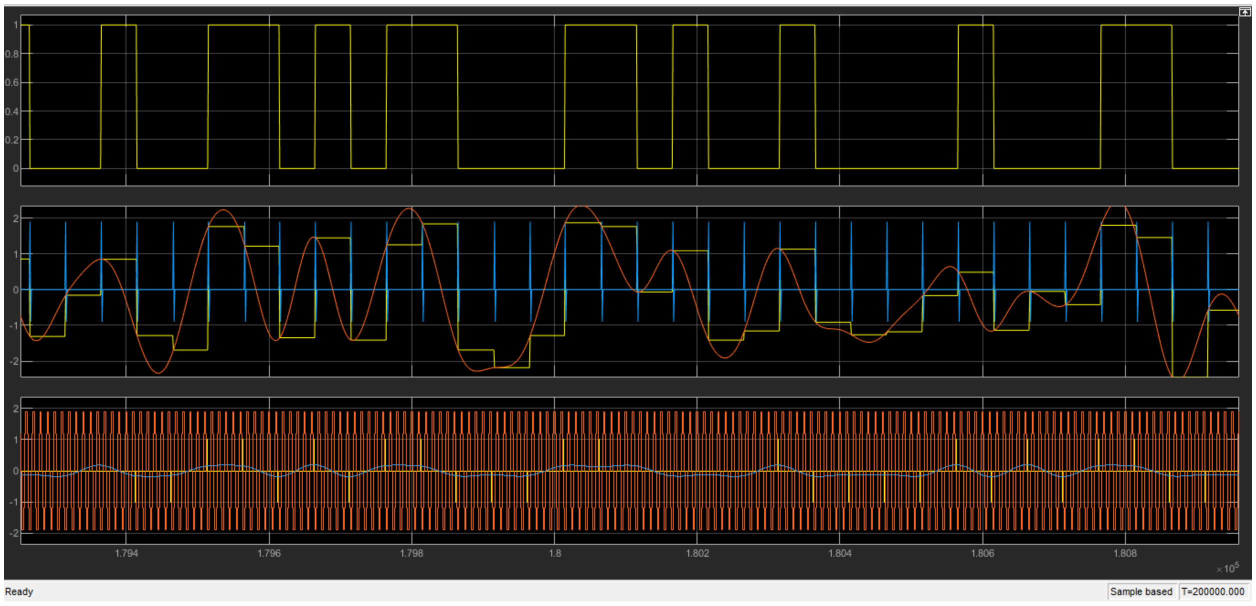


Figure 32: Receiver System Output (Noise Power: 0.01)

Lab Write-Up

1. Explain briefly how the receiver works. Also, please highlight anything that was new that you didn't know before!

We use the receiver to decode the information received from the transmitter and obtain the original baseband signal. The transmitter takes binary information, converts this into symbols, generates pulses that were weighted by complex baseband symbol weights, and up-converts this to a carrier frequency. Therefore, in order to reverse these operations, at the receiver we must do the following:

1. Multiply by a conjugate of the carrier to shift the signal back to the baseband (zero center frequency).
2. Apply a (matched) filter to remove as much noise as possible.
3. Sample at optimal points to obtain estimates of I/Q symbol weights.
4. Based on these samples, make a decision on which symbol was transmitted.
5. Convert the symbols back to bits.

2. Explain how we can get a RC response by using root filters at the transmit and receive. Explain why this is preferable to using just a single (non root) RC filter somewhere.

If we cascade two root RC filters, we get an RC shape.

Optimal operation is obtained by using identical filters at the transmitter and the receiver. In real systems where noise is present, this filter should be designed to have zero signal to noise ratio. We can get a RC response having zero ISI property by using RRC filters at the transmitter and receiver. In this way, the cascaded response is an RC response, both transmitter and receiver have identical filters and we have zero ISI.

If we only use a single RC filter instead, we do not have zero ISI and SNR would not low. Optimal operation is not obtained.

3. Explain the need for synchronization in communications systems, and how this was accomplished in this lab. Also describe what a PLL is and what it can do. What are carrier and symbol timing recovery for?

The technique to determine the suitable time for sampling of the incoming signal at the receiver block is called Time Synchronization. Carrier synchronization changes its performance based on the change in frequency and phase of the local oscillator of the received signal. When any delay, noise or disturbance is introduced in the communication channel – the phase, magnitude and delay of the received signal is affected. Therefore, synchronization is required in order to minimize any distortions as a result of external factors.

In the lab, we achieved synchronization through the use of a phase locked loop, which is a tracking algorithm for sinusoidal signals. In lab, this is used as a device that takes a sine wave as input, tracks its phase using a loop. It generates its own sine wave internally, whose phase is locked to the incoming signal. It is also free to generate any other clock or sine wave signals that are needed based on the phase. For instance, the PLL can generate a carrier with double or half the frequency, apply a constant phase shift and accomplish many other tasks.

Carrier recovery: When we transmit signals wirelessly, it is extremely difficult to make sure that the transmitted and received signals are identical. The offset can be from 10 kHz to 100 kHz in wireless RF systems. This creates a large amount of error in the receiver chain. In order to remove the carrier offset, we recover the carrier by removing modulation. Then, we lock an oscillator to the signal. Putting a pilot tone in the transmit signal is an alternative option, which provides an unmodulated carrier whose phase and frequency are directly related to the modulated transmit signal. By isolating the carrier, we can generate the carrier for the modulated signal. This is called Carrier Recovery.

Symbol Timing Recovery: In a real situation, the symbol clocks at transmitter and receiver are not the same, and we don’t know how much relative delay there is between transmitter and receiver. Symbol timing recovery is implemented by multiplying the baseband with a copy of itself shifted by half of the symbol time. This creates a sinusoid that has a frequency equal to the symbol rate, where zero-crossings of the sinusoid happen at optimal sample points. A PLL is then used to generate a sine wave locked to this carrier. The delay block allows the timing to be slightly adjusted as needed. The system flow can be seen below:

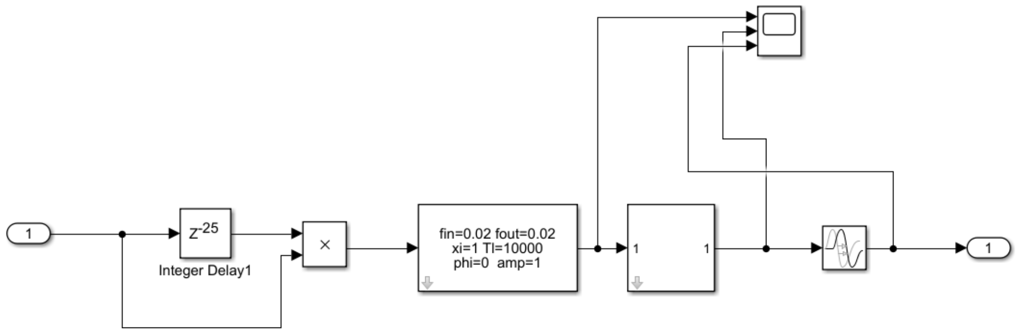


Figure 33: Symbol Timing Recovery System Flow

The signal status at different stages of the system is recorded at the scope, shown below:

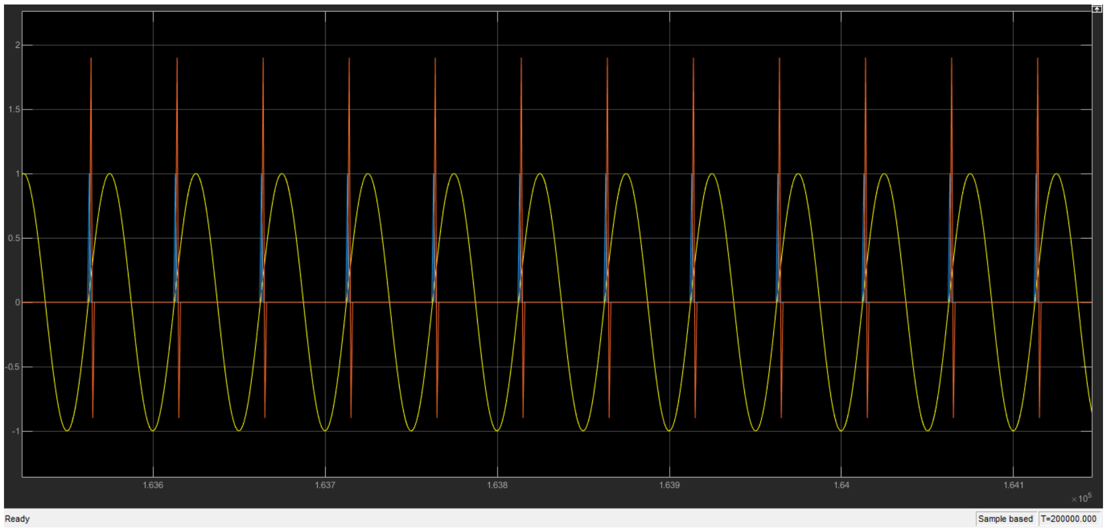


Figure 34: Symbol Timing Recovery Scope Output

We check for positive zero crossing of the sine wave, which is where we sample. The block creates a train of pulses at optimal points. This is called symbol timing recovery.

4. Give plots showing the output of your communications system for two cases: (i) with perfect ideal synchronization, and (ii) with the synchronization blocks. Circle and label points in the printout to "prove" to the TA that it is working.

(i) with perfect ideal synchronization

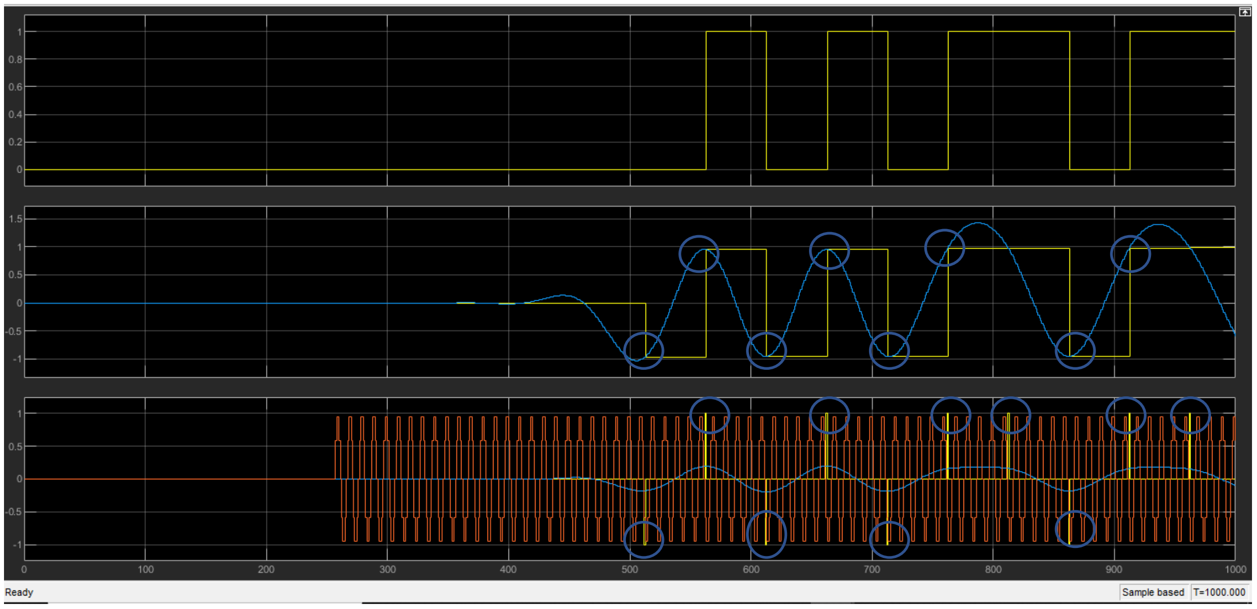


Figure 35: perfect ideal synchronization

(ii) with the synchronization blocks

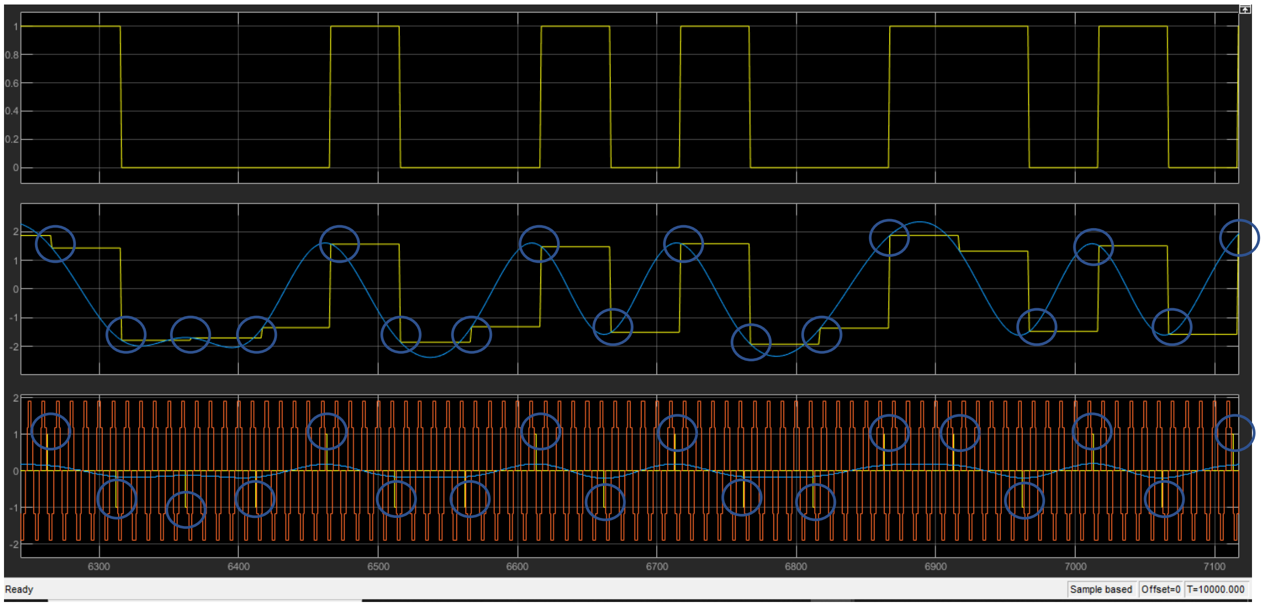


Figure 36: Synchronization with blocks

5. Describe any difficulties you experienced in getting your design to work and how you fixed these problems.

I hit a roadblock in trying to implement the simple PLL design. I talked to the TA and he confirmed that all the settings looked fine but the system was providing the wrong results. After spending a long time working on it, I asked a friend how she solved it and she mentioned that I might have forgotten to change the solver selection and solver settings, and it turned out to be the key to getting the system running properly. After that, everything looked straight-forward and the system seemed to be running smoothly.

Conclusion

Working on this lab was very fulfilling, as we learned numerous important and interesting topics in this lab. Firstly, we learned how to make a subsystem and were able to create a subsystem for our transmitter. Then, we made a simple BPSK receiver. We used a RECT Filter for our design and observed the output. Then we replaced the RECT filter with a RRC Matched Filter, and observed the differences in the output. Required modifications were made to obtain optimal performance for the matched filter. We added a pilot tone to the transmitter, and then used a filter at the receiver to implement carrier recovery. Then, we used a PLL device to track the phase of the input using a loop. Then we used Symbol Timing Recovery to find the optimal points at which to sample. Lastly, we added some noise to our system to simulate a real-world scenario, and observed how our system performed at different noise powers.

References

- Communications Basics Lab Manual: Receiver Simulink Study